Linear Models of Classification: Probabilistic Generative and Discriminative Models

Prof. Nicholas Zabaras
Center for Informatics and Computational Science
https://cics.nd.edu/
University of Notre Dame
Notre Dame, Indiana, USA

Email: nzabaras@gmail.com
URL: https://www.zabaras.com/

February 14, 2019

Contents

- Following closely Chris Bishops' PRML book, Chapter 4.
- K. Murphy, Machine Learning: A probabilistic Perspective, Chapter 8

Probabilistic Generative Models

- Models with linear decision boundaries arise from simple assumptions about the probabilistic distribution of the data.

- We here adopt **a generative approach:** we model the class-conditional densities $p(\boldsymbol{x}|C_k)$, as well as the class priors $p(C_k)$, and then use these to compute posterior probabilities $p(C_k|\boldsymbol{x})$ through Bayes' theorem.

- Let us consider two classes:

$$p(C_1 \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_1)\,p(C_1)}{p(\boldsymbol{x} \mid C_1)\,p(C_1) + p(\boldsymbol{x} \mid C_2)\,p(C_2)} = \frac{1}{1+e^{-a}} = \sigma(a)$$

where:

$$a = \ln \frac{p(\boldsymbol{x} \mid C_1)\,p(C_1)}{p(\boldsymbol{x} \mid C_2)\,p(C_2)} \quad \text{and} \quad \sigma(a) = \frac{1}{1+e^{-a}}$$

Logistic sigmoid

Logistic Sigmoid (S-shaped)

- We can approximate the logistic sigmoid function with the scaled inverse probit function.
- The derivatives of the two curves are the same for $a = 0$ (selection of $\lambda^2 = \frac{\pi}{8}$).

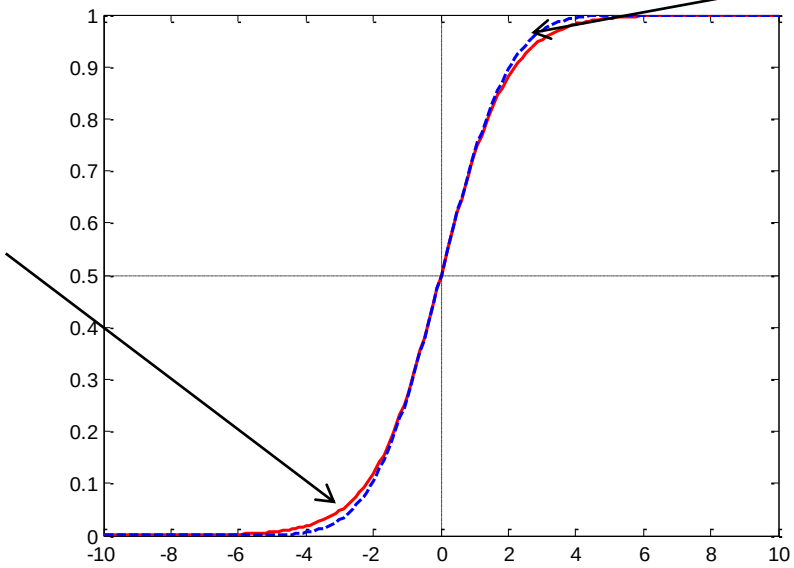
$$\Phi(\lambda a) = \int_{-\infty}^{\lambda a} \mathcal{N}(\theta \mid 0,1)\, d\theta$$

-- dashed line

Scaled inverse Probit function

Logistic sigmoid

$$\sigma(a) = \frac{1}{1+e^{-a}}$$



MatLab Code

- The logistic sigmoid function satisfies: $\sigma(-a) = 1 - \sigma(a)$
- The *inverse of $\sigma(a)$ is the logit function*: $a = \ln \frac{\sigma}{1-\sigma} = \ln \frac{p(C_1 \mid \boldsymbol{x})}{p(C_2 \mid \boldsymbol{x})}$

Probabilistic Generative Models

- The appearance of the logistic sigmoid is rather arbitrary up to this point.

$$p(C_1 \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_1)p(C_1)}{p(\boldsymbol{x} \mid C_1)p(C_1) + p(\boldsymbol{x} \mid C_2)p(C_2)} = \frac{1}{1+e^{-a}} = \sigma(a)$$

- This will be useful provided $a(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x} \mid C_1)p(C_1)}{p(\boldsymbol{x} \mid C_2)p(C_2)}$ takes a simple functional form.

- We will consider problems with $a(\boldsymbol{x})$ being a linear function of $\boldsymbol{x}$.

- In this case, the posterior probability is governed by a generalized linear model.

Probabilistic Generative Models $K > 2$

- For the case of $K > 2$ *classes, we have*

$$p(C_k \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_k)\, p(C_k)}{\sum_j p(\boldsymbol{x} \mid C_j)\, p(C_j)} = \frac{e^{a_k}}{\sum_j e^{a_j}}, \; a_k = \ln\left(p(\boldsymbol{x} \mid C_k)\, p(C_k) \right)$$

- This is known as the normalized exponential and can be regarded as a multiclass generalization of the logistic sigmoid.
- Here we define:

$$a_k = \ln\left(p(\boldsymbol{x} \mid C_k)\, p(C_k) \right)$$

- The normalized exponential is known as the softmax function.
- It can be seen as a smoothed version of the max function because, if $a_k \geq a_j$ for all $j \neq k$, then $p(C_k|\boldsymbol{x}) \cong 1$, and $p(C_j|\boldsymbol{x}) \cong 0$.

Gaussian Class Conditionals

- Assume that the class-conditional densities are Gaussians and all classes share the same covariance matrix:

$$p(\boldsymbol{x} \mid C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right\}$$

- We consider the case of two classes:

$$p(C_1 \mid \boldsymbol{x}) = \sigma(a), \; a(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x} \mid C_1)p(C_1)}{p(\boldsymbol{x} \mid C_2)p(C_2)}$$

where:

$$a(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) + \ln \frac{p(C_1)}{p(C_2)} =$$

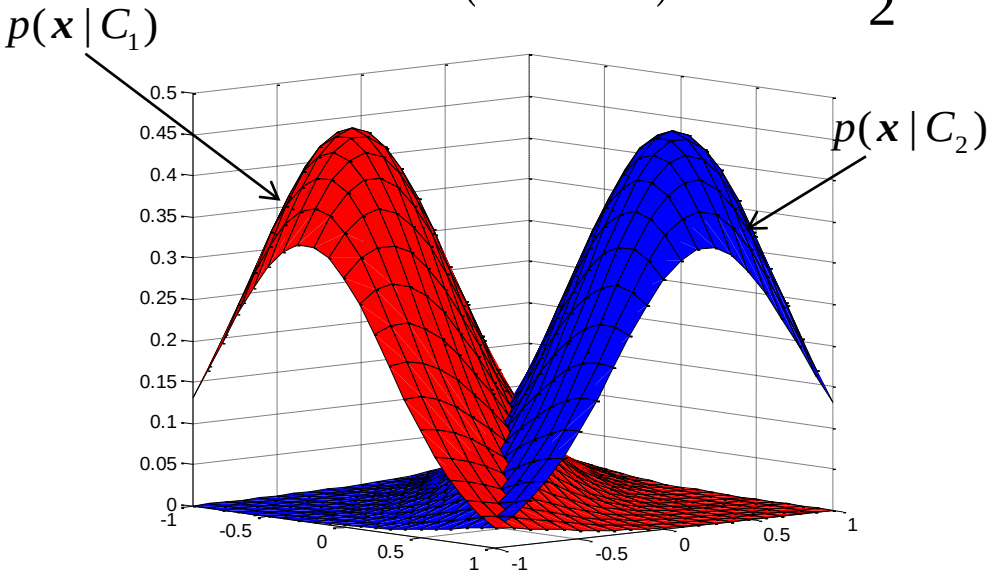
$$= \underbrace{\left(\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} - \boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1}\right)}_{\boldsymbol{w}^T} \boldsymbol{x} \underbrace{-\frac{1}{2}\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln \frac{p(C_1)}{p(C_2)}}_{w_0}$$

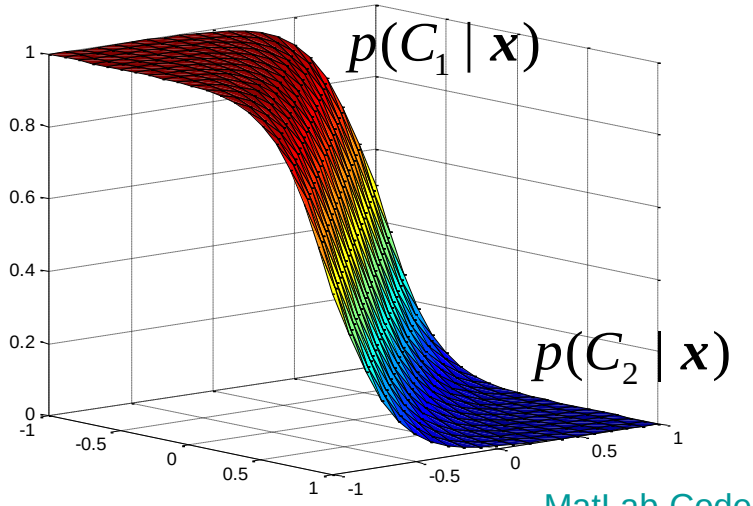
Gaussian Class Conditionals and Posteriors

- We thus obtain a linear function of $\boldsymbol{x}$:

$$p(C_1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x} + w_0)$$

$$\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right),\ w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(C_1)}{p(C_2)}$$





MatLab Code

- Class conditional densities (left) and corresponding posteriors (right).
- On the right, the surface is colored using a proportion of red ink given by $p(C_1|\boldsymbol{x})$ and a proportion of blue ink given by $p(C_2|\boldsymbol{x}) = 1 - p(C_1|\boldsymbol{x})$

Gaussian Class Posteriors

$$p(C_1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x} + w_0)$$

- *The decision boundaries correspond to surfaces along which the posterior probabilities $p(C_k|\boldsymbol{x})$ are constant*.

- They are linear in input space.

- The prior probabilities $p(C_k)$ enter only through the bias parameter w_0

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln \frac{p(C_1)}{p(C_2)}$$

so that *changing $p(C_k)$ only leads to parallel shifts of the decision boundaries.*

Probabilistic Generative Models K>2

- For the case of $K > 2$ classes, we have

$$p(C_k \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_k)\, p(C_k)}{\sum_j p(\boldsymbol{x} \mid C_j)\, p(C_j)} = \frac{e^{a_k}}{\sum_j e^{a_j}} \quad \text{where } a_k = \ln\left(p(\boldsymbol{x} \mid C_k)\, p(C_k) \right)$$

- The resulting decision boundaries, corresponding to the minimum misclassification rate, occur when the two largest posterior probabilities are equal, and so are defined by linear functions of $\boldsymbol{x}$, leading to a generalized linear model.

- For this case, we can see immediately that:

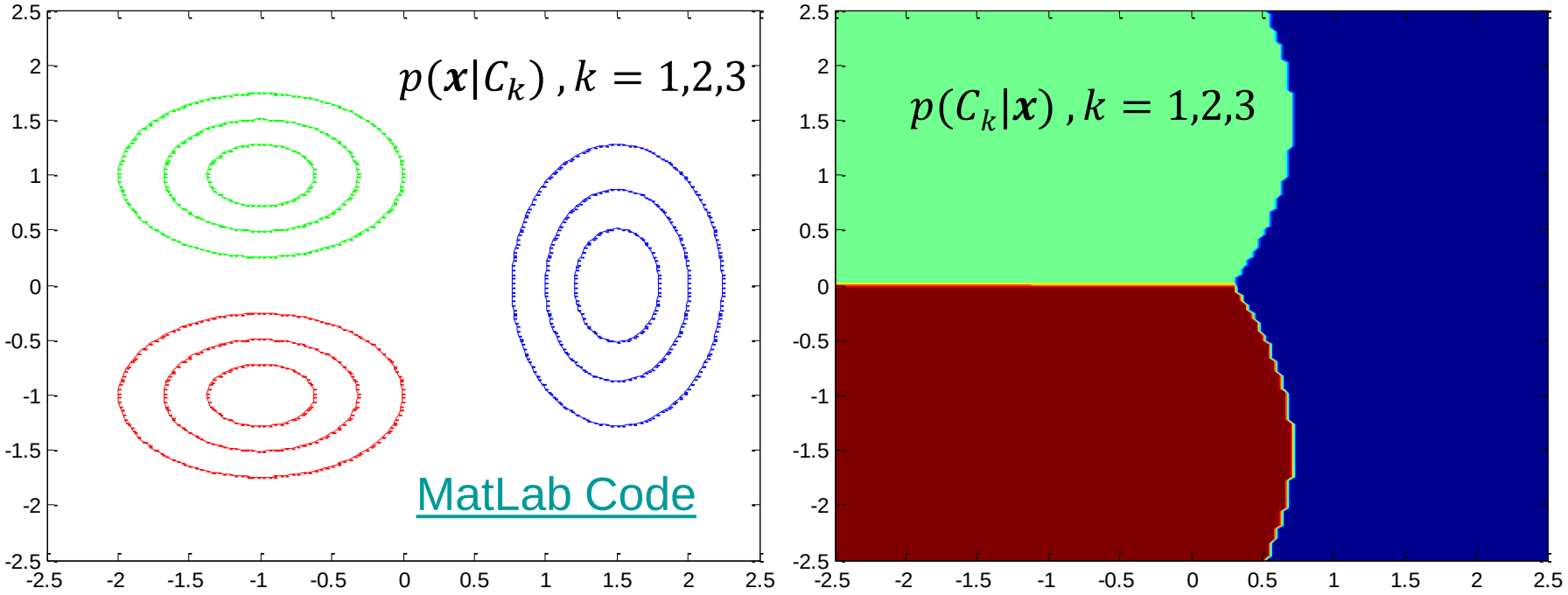
$$a_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}$$

$$\boldsymbol{w}_k = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k \,,\; w_{k0} = -\frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \ln p(C_k)$$

- $a_k(\boldsymbol{x})$ are linear functions of $\boldsymbol{x}$ due to the cancellation of the quadratic terms with the shared $\boldsymbol{\Sigma}$.

Probabilistic Generative Models $K > 2$

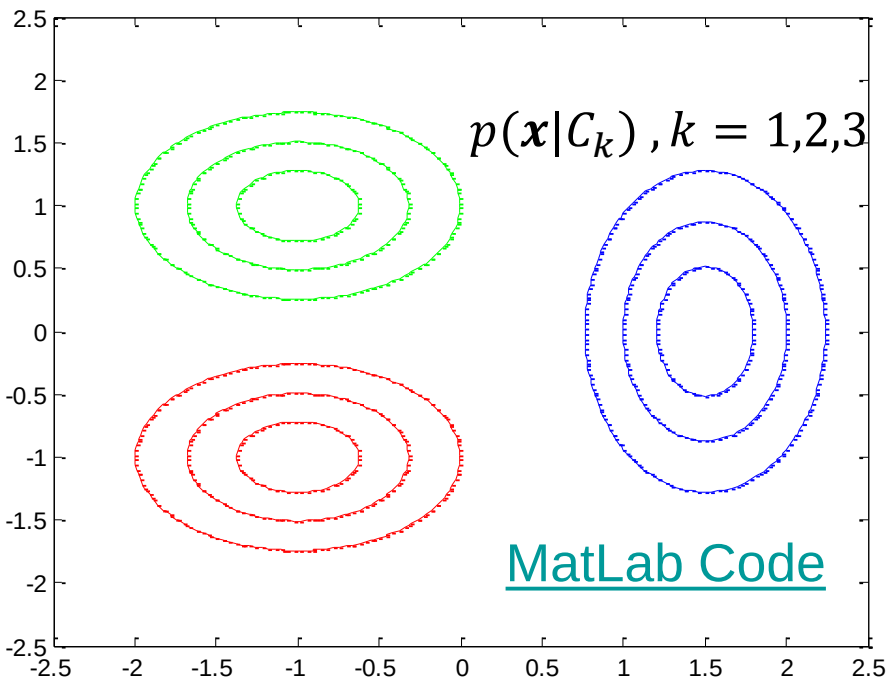
- If we relax the assumption of a shared covariance matrix and allow each $p(\boldsymbol{x}|C_k)$ to have its own $\boldsymbol{\Sigma}_k$, then we obtain quadratic functions of $\boldsymbol{x}$ and a quadratic discriminant.

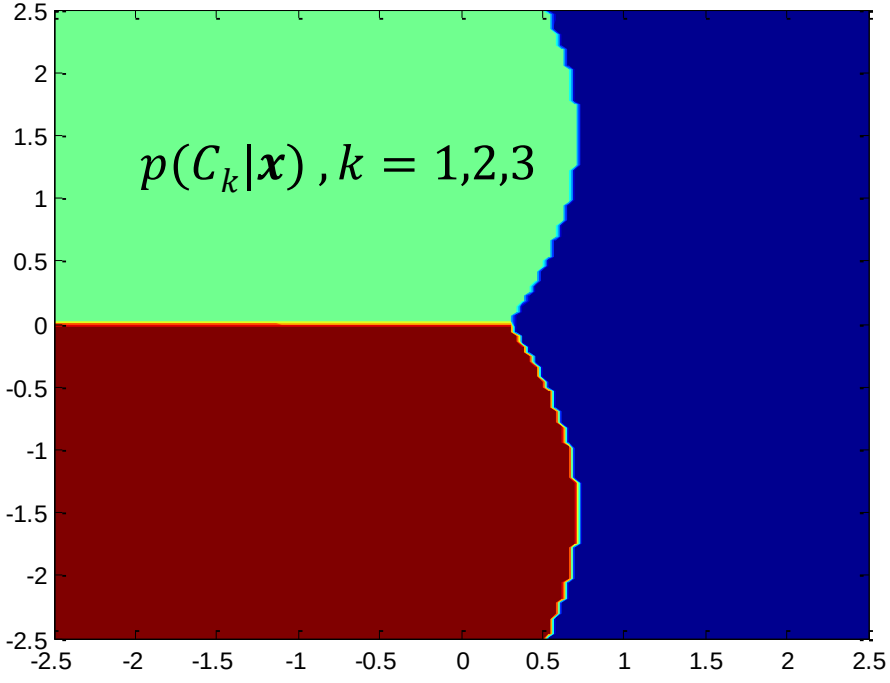


- The left-hand plot shows $p(\boldsymbol{x}|C_k), k = 1,2,3$ each a Gaussian distribution. The red & green classes have the same $\boldsymbol{\Sigma}$.

Probabilistic Generative Models $K > 2$

- The right-hand plot shows the posteriors $p(C_k|\boldsymbol{x})$, $k = 1,2,3$, in which the three colors represent the posterior probabilities for the respective 3 classes (scaled).
- Note that for *the two classes with the same covariance, the decision boundary is linear while for the other 2 is quadratic*.





Maximum Likelihood Solution

- Once we have a parametric form for $p(\boldsymbol{x}|C_k)$, we can compute the parameters, together with the prior class probabilities $p(C_k)$, using MLE.
- This requires a training data set comprising of $\boldsymbol{x}$ and their corresponding class labels.
- Consider the case of two classes each having a Gaussian class-conditional density with a shared covariance matrix.
- Suppose we have a data set $\{\boldsymbol{x}_n, t_n\}$ where $n = 1, \ldots, N$. Here $t_n = 1$ denotes class C_1 and $t_n = 0$ denotes C_2.
- We denote the prior class probabilities $p(C_1) = \pi$, $p(C_2) = 1 - \pi$.

Maximum Likelihood Solution

- For a data point $\boldsymbol{x}_n$ from class C_1, we have $t_n = 1$ and hence

$$p(\boldsymbol{x}_n, C_1) = p(C_1)\, p(\boldsymbol{x}_n \mid C_1) = \pi \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$

- Similarly for class C_2, we have $t_n = 0$

$$p(\boldsymbol{x}_n, C_2) = p(C_2)\, p(\boldsymbol{x}_n \mid C_2) = \left(1-\pi\right) \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

- Thus *the likelihood function is given by*

$$p(\boldsymbol{t}, \boldsymbol{X} \mid \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma})\right]^{t_n} \left[\left(1-\pi\right) \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma})\right]^{1-t_n}$$

where $\boldsymbol{t} = (t_1, \ldots, t_N)^T$.

- We maximize the log likelihood function.

Maximum Likelihood Solution

$$p(\boldsymbol{t}, \boldsymbol{X} \mid \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \right]^{t_n} \left[(1-\pi) \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \right]^{1-t_n}$$

- Consider first the maximization with respect to π. The terms of interest in the log-likelihood are:

$$\sum_{n=1}^{N} \{ t_n \ln \pi + (1-t_n) \ln(1-\pi) \}$$

- Taking derivative wrt π and setting it equal to zero:

$$\pi = \frac{1}{N} \sum_{n=1}^{N} t_n = \frac{N_1}{N_1 + N_2}$$

where N_i the total number of data points in class C_i.

- As expected, the maximum likelihood estimate for π is simply the fraction of points in class C_1.
- This result is easily generalized to the multiclass case (see next) where the MLE of the prior probability associated with class C_k is given by the fraction of the training set points N_k.

Maximum Likelihood Solution: Multiclass

$$p(\{\boldsymbol{\phi}_n, \boldsymbol{t}_n\} \mid \{\pi_k\}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \left[\pi_k \, p(\boldsymbol{\phi}_n \mid C_k)\right]^{t_{nk}}$$

- For the multiclass case, the log likelihood takes the form:

$$\ln p(\{\boldsymbol{\phi}_n, \boldsymbol{t}_n\} \mid \{\pi_k\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\ln \pi_k + \ln p(\boldsymbol{\phi}_n \mid C_k)\right), \quad \sum_{k=1}^{K} \pi_k = 1$$

- To maximize the log likelihood, we consider:

$$\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\ln \pi_k + \ln p(\boldsymbol{\phi}_n \mid C_k)\right) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1\right)$$

- Taking the derivative wrt π_k we arrive at the same result as for two classes:

$$\sum_{n=1}^{N} \frac{t_{nk}}{\pi_k} + \lambda = 0 \Rightarrow \pi_k \lambda = -\sum_{n=1}^{N} t_{nk} = -N_k \Rightarrow \begin{cases} \sum_{k=1}^{K} \pi_k \lambda = -N \\ \pi_k \lambda = -N_k \end{cases} \Rightarrow \begin{cases} \lambda = -N \\ \pi_k \lambda = -N_k \end{cases} \Rightarrow \pi_k = \frac{N_k}{N}$$

Maximum Likelihood Solution

$$p(\boldsymbol{t}, \boldsymbol{X} \mid \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \right]^{t_n} \left[(1-\pi) \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \right]^{1-t_n}$$

- Consider now the maximization with respect to μ_1. The terms of interest in the log-likelihood are:

$$\sum_{n=1}^{N} t_n \ln \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2} \sum_{n=1}^{N} t_n (\boldsymbol{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_1) + const.$$

- Taking derivative wrt μ_1 and setting it equal to zero:

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^{N} t_n \boldsymbol{x}_n, \text{ and similarly } \boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^{N} (1-t_n) \boldsymbol{x}_n$$

- Thus the MLE for μ_i is simply the mean of the fraction of points in class C_i.
- This result is easily generalized to the multiclass case. The MLE of the prior probability associated with class C_k is given by the fraction of the training set points N_k.

Maximum Likelihood Solution

$$p(\mathbf{t} \mid \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \right]^{t_n} \left[(1-\pi) \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \right]^{1-t_n}$$

- Consider finally the maximization with respect to $\boldsymbol{\Sigma}$. The terms of interest in the log-likelihood are:

$$-\frac{1}{2}\sum_{n=1}^{N} t_n \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1)$$

$$-\frac{1}{2}\sum_{n=1}^{N} (1-t_n) \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} (1-t_n)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_2)$$

$$= -\frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{N}{2} Tr\left(\boldsymbol{\Sigma}^{-1}\mathbf{S}\right), \text{ where } \mathbf{S} = \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2$$

$$\mathbf{S}_1 = \frac{1}{N_1}\sum_{n \in C_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T, \quad \mathbf{S}_2 = \frac{1}{N_2}\sum_{n \in C_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T$$

- Using the result for the MLE for a Gaussian distribution, we obtain $\boldsymbol{\Sigma} = \mathbf{S}$. This result is a weighted average of the covariance matrices associated with each class separately.

Maximum Likelihood Solution

$$\pi = \frac{N_1}{N_1 + N_2}, \ \boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n, \boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1 - t_n)\mathbf{x}_n$$

$$\boldsymbol{\Sigma} = \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2, \ \mathbf{S}_1 = \frac{1}{N_1}\sum_{n \in C_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T, \mathbf{S}_2 = \frac{1}{N_2}\sum_{n \in C_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T$$

- These results are easily extended to the K class problem to obtain the corresponding MLE solutions for the parameters (for Gaussian class-conditionals with a shared covariance matrix).
- As we have seen before, MLE is not robust to outliers.
  - Fitting Gaussian distributions to the classes is not robust to outliers, because the MLE of a Gaussian is not robust.

Discrete Features: Naive Bayes Assumption

- Here we consider the case of discrete feature values x_i. Consider binary feature values $x_i \in \{0, 1\}, i = 1, \ldots, D$.
- For D inputs, a general distribution corresponds to a table of 2^D numbers for each class, containing $2^D - 1$ independent variables (due to the summation constraint).
- This grows exponentially with the number D of features.
- Consider the Naive Bayes Assumption: the feature values are treated as independent, conditioned on the class C_k.
- Thus we have class-conditional distributions of the form

$$p(\mathbf{x} \mid C_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i} (1 - \mu_{ki})^{1 - x_i}$$

Discrete Features

$$p(\boldsymbol{x} \mid C_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i} (1-\mu_{ki})^{1-x_i}$$

- There are D independent parameters for each class.
- Here μ_{ki} is the probability the i^{th} feature takes the value 1.
- Substituting into $a_k = \ln\big(p(\boldsymbol{x} \mid C_k)\, p(C_k)\big)$ gives:

$$a_k(\boldsymbol{x}) = \sum_{i=1}^{D} \big\{ x_i \ln \mu_{ki} + (1-x_i) \ln (1-\mu_{ki}) \big\} + \ln p(C_k)$$

- These are linear functions of the input values x_i.
- For $K = 2$, we can alternatively consider the logistic sigmoid

$$p(C_1 \mid \boldsymbol{x}) = \sigma(a), \; a(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x} \mid C_1)\, p(C_1)}{p(\boldsymbol{x} \mid C_2)\, p(C_2)}$$

- Analogous results are obtained for discrete variables each of which can take $M > 2$ states.

Exponential Family

- We have have seen that for both Gaussian and discrete inputs, the posterior class probabilities are given by generalized linear models with logistic sigmoid ($K = 2$) or softmax ($K \geq 2$ classes) activation functions.

- This result is general for the class-conditional densities $p(\boldsymbol{x}|C_k)$ that are members of the exponential family.

- For the exponential family, the distribution of $\boldsymbol{x}$ can be written in the form

$$p(\boldsymbol{x} \mid \boldsymbol{\lambda}_k) = h(\boldsymbol{x}) g(\boldsymbol{\lambda}_k) \exp\left\{\boldsymbol{\lambda}_k^T u(\boldsymbol{x})\right\}$$

- Here, we consider the case $u(\boldsymbol{x}) = \boldsymbol{x}$.

Exponential Family

$$p(\boldsymbol{x} \mid \boldsymbol{\lambda}_k) = h(\boldsymbol{x}) g(\boldsymbol{\lambda}_k) \exp\left\{\boldsymbol{\lambda}_k^T u(\boldsymbol{x})\right\}$$

- With $u(\boldsymbol{x}) = \boldsymbol{x}$ and by introducing a scaling parameter s as $p(x \mid s) = (1/s) f(x/s)$, we obtain the restricted set of exponential family class-conditional densities of the form:

$$p(\boldsymbol{x} \mid \boldsymbol{\lambda}_k, s) = \frac{1}{s} h(\frac{1}{s}\boldsymbol{x}) g(\boldsymbol{\lambda}_k) \exp\left\{\frac{1}{s}\boldsymbol{\lambda}_k^T \boldsymbol{x}\right\}$$

- Here each class has its own parameter λ_k but all classes share the same scale parameter s.

- For the 2 class problem, we substitute this expression for the class-conditional densities into

$$a(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x} \mid C_1) p(C_1)}{p(\boldsymbol{x} \mid C_2) p(C_2)}$$

Exponential Family

$$p(\boldsymbol{x} \mid \boldsymbol{\lambda}_k, s) = \frac{1}{s} h(\frac{1}{s}\boldsymbol{x}) g(\boldsymbol{\lambda}_k) \exp\left\{\frac{1}{s}\boldsymbol{\lambda}_k^T \boldsymbol{x}\right\} \qquad a(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x} \mid C_1) p(C_1)}{p(\boldsymbol{x} \mid C_2) p(C_2)}$$

- The posterior class probability is again given by a logistic sigmoid acting on a linear function $a(\boldsymbol{x})$

$$p(C_1 \mid \boldsymbol{x}) = \sigma(a(\boldsymbol{x}))$$

which is given by

$$a(\boldsymbol{x}) = \frac{1}{s}\left(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\right)^T \boldsymbol{x} + \ln g(\boldsymbol{\lambda}_1) - \ln g(\boldsymbol{\lambda}_2) + \ln p(C_1) - \ln p(C_2)$$

- Similarly for the K –class problem, we substitute the class-conditionals directly in $a_k(\boldsymbol{x}) = \ln\left(p(\boldsymbol{x} \mid C_k) p(C_k)\right)$ to obtain:

$$a_k(\boldsymbol{x}) = \frac{1}{s}\boldsymbol{\lambda}_k^T \boldsymbol{x} + \ln g(\boldsymbol{\lambda}_k) + \ln p(C_k)$$

which is linear in $\boldsymbol{x}$.

Probabilistic Discriminative Models

- *For the two-class problem*, *the posterior probability of class* C_k *can be written as a logistic sigmoid* acting on a linear function of $\boldsymbol{x}$, for a wide choice of $p(\boldsymbol{x}|C_k)$.
- *For the multiclass problem*, the posterior probability of C_k is given by *a softmax transformation* of a linear function of $\boldsymbol{x}$.
- *For specific choices of* $p(\boldsymbol{x}|C_k)$*, we used MLE to determine the parameters of the densities and the class priors* $p(C_k)$ *and then used Bayes' theorem to find the posterior class probabilities.*
- An alternative approach that we follow in this lecture is to *use the functional form of the generalized linear model explicitly and determine its parameters directly by using MLE*.
- There is an efficient algorithm finding such solutions known as iterative reweighted least squares, or *IRLS.*

Probabilistic Discriminative Models

- Indirect approach – Generative Modeling: find the parameters of the generalized linear model, by fitting class-conditional densities and class priors separately and then applying Bayes' theorem.
  - With this model, we *can generate synthetic data by drawing values of $\boldsymbol{x}$ from the marginal $p(\boldsymbol{x})$.*
- Direct approach – Discriminative Modeling: we are maximizing a likelihood function defined through $p(C_k|\boldsymbol{x})$, which represents a form of discriminative training.
- An advantage of the discriminative approach is that there are typically fewer adaptive parameters to be determined.
- It leads to improved predictive performance when the class-conditional density assumptions are a poor approximation to the true distributions.

Fixed Basis Functions

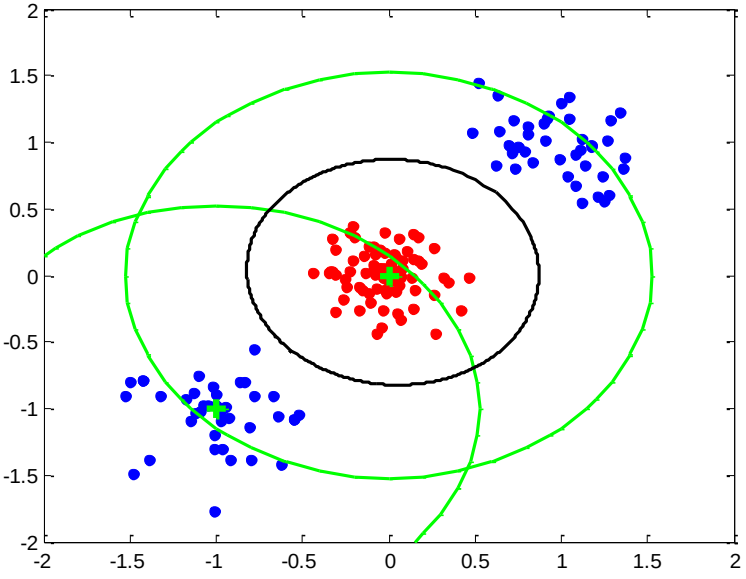
- First make a fixed nonlinear transformation of the inputs using a vector of basis functions $\phi(\boldsymbol{x})$, then apply the algorithms discussed up to now.

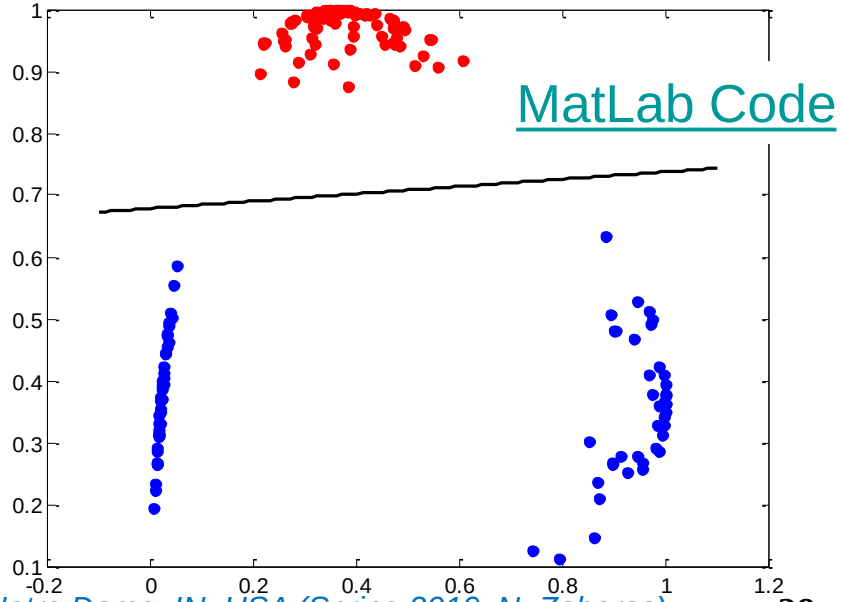
- The resulting decision boundaries will be linear in the feature space ϕ, and these correspond to nonlinear decision boundaries in the original $\boldsymbol{x}$ space.

- Classes that are linearly separable in the feature space $\phi(\boldsymbol{x})$ need not be linearly separable in the original input space $\boldsymbol{x}$.

Nonlinear Basis in Classification Models

- Left: original space (x_1, x_2) with data from red & blue classes
- 2 'Gaussian' basis functions $\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x})$ are defined in (x_1, x_2) with green crosses the centers & green circles the contours.
- Right: the corresponding feature space (ϕ_1, ϕ_2) together with the linear decision boundary obtained by a logistic regression model. This corresponds to a nonlinear decision boundary in (x_1, x_2) (black curve on the left).





MatLab Code

Fixed Basis Functions

- One of the basis functions is typically set to a constant, say $\phi_0(x) = 1$, so that w_0 plays the role of a bias.
- For problems with significant overlap between the class-conditional densities $p(\boldsymbol{x}|C_k)$, the posterior probabilities $p(C_k|\boldsymbol{x})$ at least for some values of $\boldsymbol{x}$ may not be 0 or 1.
  - In such cases, the optimal solution is obtained by modeling the posterior probabilities accurately and then applying standard decision theory.
- The transformation $\phi(\boldsymbol{x})$ cannot remove class overlap, it can increase the level of overlap, or create overlap where none existed in the $\boldsymbol{x}$ space.
- However, suitable choices of nonlinearity can make the process of modeling the posterior probabilities easier.

Fixed Basis Functions

- Fixed basis function models have serious limitations.
- We should allow the basis functions to adapt to the data.
- Still models with fixed nonlinear basis functions are important as they introduce many of the key concepts needed for an understanding models with data-adaptive basis.

Logistic Regression Generalized Linear Models

- Consider a simple generalized linear problem with a two-class classification.

- In generative models we saw that the posterior probability of class C_1 can be written as a logistic sigmoid acting on a linear function of the feature vector $\boldsymbol{\phi}$ so that

$$p(C_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\boldsymbol{w}^T\boldsymbol{\phi})$$

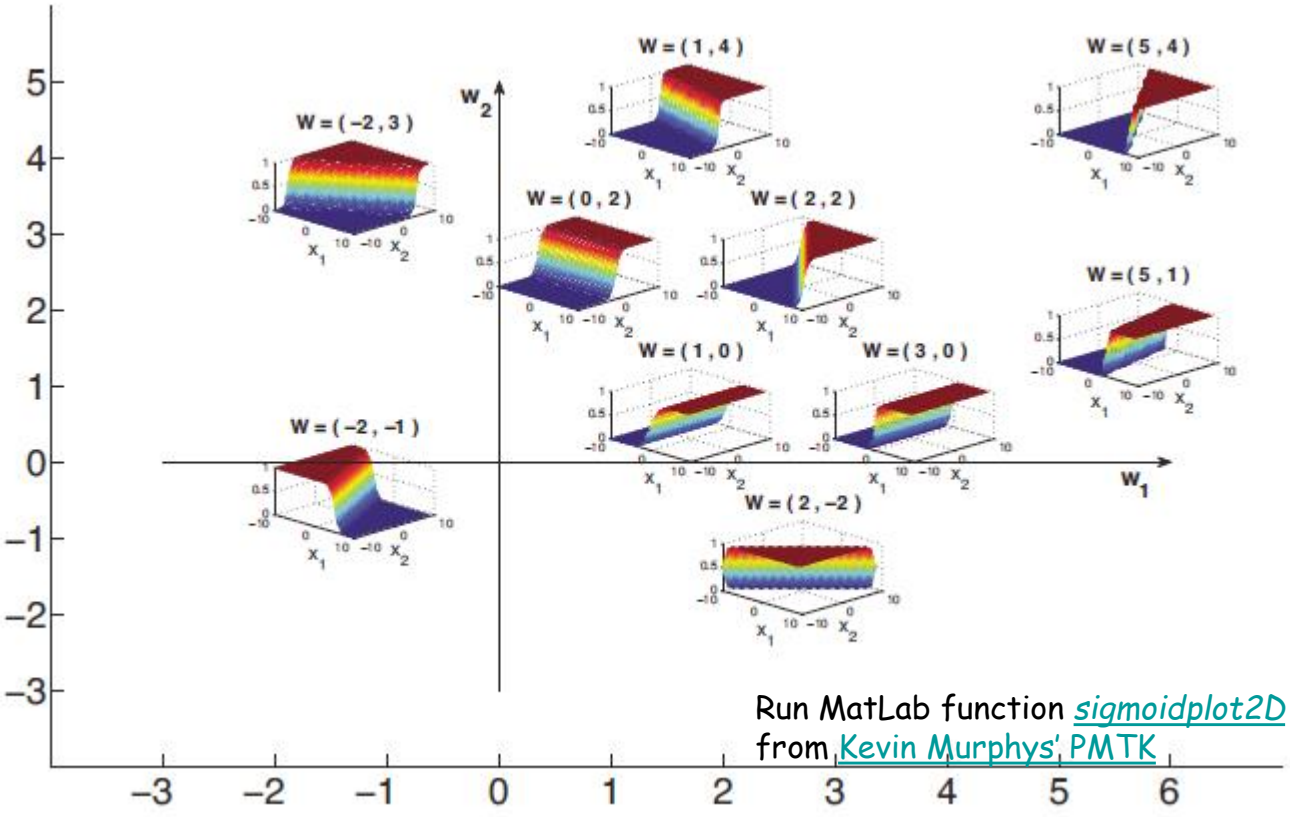
with $p(C_2|\boldsymbol{\phi}) = 1 - p(C_1|\boldsymbol{\phi})$. Here $\sigma(\cdot)$ is the logistic sigmoid function defined by

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

- This model is known as logistic regression – but *note its a model for classification not regression.*

Plots of $p(C_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma\left(w^T\boldsymbol{\phi}\right)$

- Plots of $p(C_1|f) = \sigma\left(w_1x_1 + w_2x_2\right)$ for a 2D input and different $\boldsymbol{w}$.
- If we threshhold these probabilities at 0.5, we induce a linear decision boundary whose normal is $\boldsymbol{w}$.
- To the right of this have $\sigma\left(\boldsymbol{w}^T\boldsymbol{x}\right) > 0$ and to the left $\sigma\left(\boldsymbol{w}^T\boldsymbol{x}\right) < 0$.



Logistic Regression

$$p(C_1|\boldsymbol{\phi}) = \boldsymbol{y}(\boldsymbol{\phi}) = \sigma(\boldsymbol{w}^T\boldsymbol{\phi})$$

- *For an M –dimensional feature space $\boldsymbol{\phi}$, this model has M adjustable parameters.*
- If we had fitted Gaussian class conditional densities using MLE, we would have used $2M$ parameters for the means and $M(M+1)/2$ parameters for the (shared) covariance. Together with the class prior $p(C_1)$, this gives a total of $M(M+5)/2+1$ parameters, i.e. $\mathcal{O}(M^2)$, in contrast to $\mathcal{O}(M)$ dependence on M in logistic regression.
- For large M, there is advantage in working with logistic regression.
- We now use MLE to determine the parameters in logistic regression. We will use:

$$\frac{d\sigma}{da} = \sigma(1-\sigma)$$

Logistic Regression - Cross-Entropy Error

- For a data set $\{\boldsymbol{\phi}_n, t_n\}$, where $t_n \in \{0,1\}$ *and* $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\boldsymbol{x}_n)$, with $n = 1,\ldots,N$, the likelihood function can be written

$$p(\boldsymbol{t}\,|\,\boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n},\ \boldsymbol{t} = (t_1,\ldots,t_N)^T,\ y_n = p(C_1\,|\,\boldsymbol{\phi}_n)$$

- Using this, we define the cross-entropy error function as:

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{t}\,|\,\boldsymbol{w}) = -\sum_{n=1}^{N}\left\{t_n \ln y_n + (1-t_n)\ln\left(1-y_n\right)\right\},\ y_n = \sigma(a_n),\ a_n = \boldsymbol{w}^T\boldsymbol{\phi}_n$$

- Taking the gradient wrt $\boldsymbol{w}$ and using $\dfrac{d\sigma}{da} = \sigma(1-\sigma)$ gives:

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}(y_n - t_n)\boldsymbol{\phi}_n,\ y_n = \sigma(a_n),\ a_n = \boldsymbol{w}^T\boldsymbol{\phi}_n$$

- The contribution to $\nabla E(\boldsymbol{w})$ from data point n is given by the 'error' $y_n - tn$ between the target value and the prediction of the model, times the basis $\boldsymbol{\phi}_n$. This is identical to the gradient of the sum-of-squares error function for linear regression.

Logistic Regression - Sequential Update

- We use this result

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} (y_n - t_n)\boldsymbol{\phi}_n,\ y_n = \sigma(a_n),\ a_n = \boldsymbol{w}^T\boldsymbol{\phi}_n$$

  to derive a sequential algorithm in which data are presented one at a time.

- The weight vector is updated using

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta\nabla E_n = \boldsymbol{w}^{(\tau)} - \eta(y_n - t_n)\boldsymbol{\phi}_n$$

- Note that for linearly separable data (the hyperplane $\sigma = 0.5, \boldsymbol{w}^T\boldsymbol{\phi} = 0$ separates the two classes) any decision boundary separating the two classes needs to satisfy:

$$\boldsymbol{w}^T\boldsymbol{\phi}_n \begin{cases} \geq 0 \ if\ t_n = 1 \\ < 0 \ otherwise \end{cases}$$

- However we can see from the top Eq. that the negative log likelihood is minimized when: $y_n = \sigma\left(\boldsymbol{w}^T\boldsymbol{\phi}_n\right) = t_n, n = 1,\ldots,N$. This requires that the sigmoid function saturates which occurs when its argument

$$\boldsymbol{w}^T\boldsymbol{\phi}_n \to \pm\infty, i.e.\ when \|\boldsymbol{w}\| \to \infty.$$

Logistic Regression Linearly Separable Data

- MLE can exhibit severe over-fitting for data sets *that are linearly separable.*
- Indeed for linearly separable data, the hyperplane corresponding to $P(C_1|\boldsymbol{x}) = \sigma = 0.5 \rightarrow \boldsymbol{w}^T\boldsymbol{\phi} = 0$ separates the two classes and will have the following:

$$\boldsymbol{w}^T\phi_n = \begin{cases} \geq 0 \; if \; t_n = 1 \\ < 0 \; if \; t_n = 0 \end{cases}$$

- In addition we can see that the negative log-likelihood is minimized when

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}(y_n - t_n)\boldsymbol{\phi}_n = 0 \Rightarrow y_n = \sigma(\boldsymbol{w}^T\boldsymbol{\phi}_n) = t_n \; \forall n, \; i.e. \; y_n = t_n = 0 \text{ or } y_n = t_n = 1$$

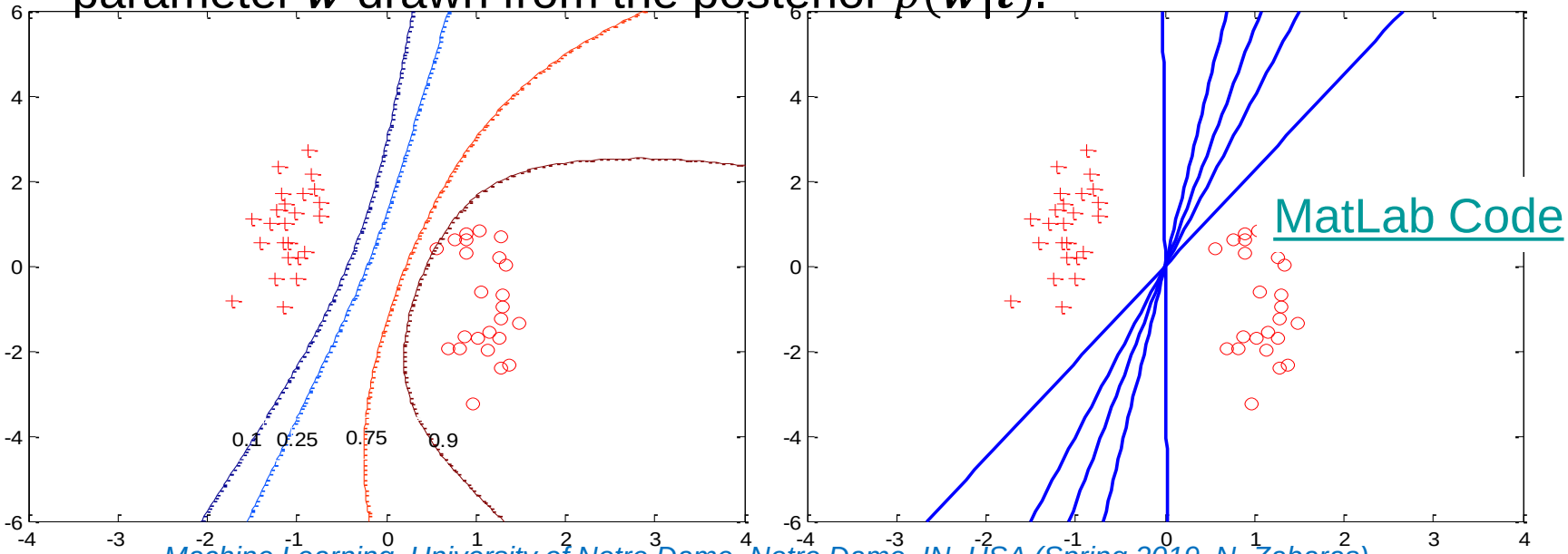
- This occurs when *the sigmoid function is saturated*, which occurs when its argument $\boldsymbol{w}^T\boldsymbol{\phi}$ goes to $\pm\infty$, i.e. *when the magnitude of* $\boldsymbol{w}$ *goes to* ∞.
- The logistic sigmoid function becomes infinitely steep in feature space (step function) so every training point from each class *k is* assigned a posterior probability $p(C_k|\boldsymbol{x}) = 1$.
- There are infinite solutions since any separating hyperplane will give the same posterior probabilities at the training data points.

Logistic Regression Linearly Separable Data

- Maximum likelihood provides no way to favor one such solution over another.
- *Which solution is found depends on the choice of optimization algorithm and on the parameter initialization.*
- The problem arises even if the number of data points is large compared to the number of parameters in the model, so long as the training data set is linearly separable.
- The singularity can be avoided by inclusion of a prior and finding a MAP solution for $\boldsymbol{w}$, or equivalently by adding a regularization term to the error function.

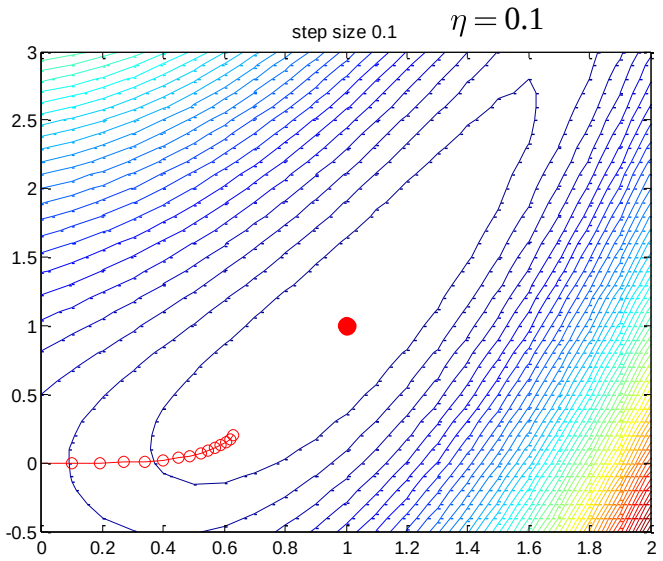
Logistic Regression Linearly Separable Data

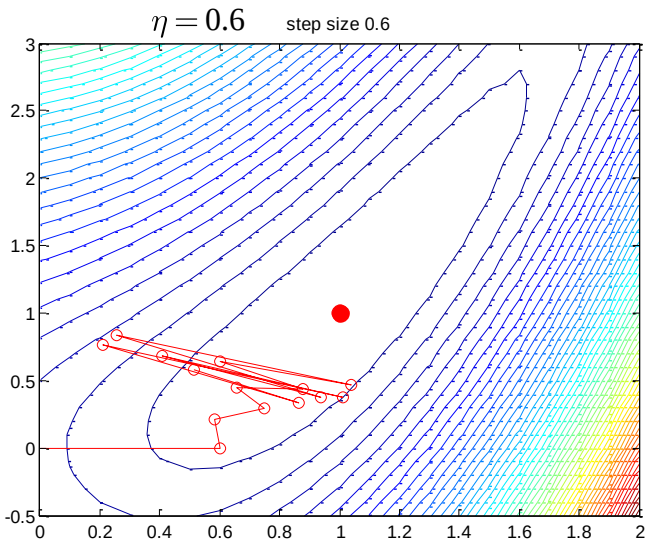
- Bayesian approach to logistic regression for linearly separable data set.
- Left: the predictive distribution using Variational Inference (details will be given in another lecture). The decision boundary lies between the clusters of data points. The contours of the predictive distribution splay out away from the data reflecting the greater uncertainty in the classification of such regions.
- Right: the decision boundaries corresponding to 5 samples of the parameter $\boldsymbol{w}$ drawn from the posterior $p(\boldsymbol{w}|\boldsymbol{t})$.



Gradient Descent Example

- The effect of the learning rate η on the performance of a steepest descent algorithm is shown on a simple function.





$$f(\boldsymbol{\theta}) = 0.5\ \theta_1^2 - \theta_2\ ^2 + 0.5\ \theta_1 - 1\ ^2$$

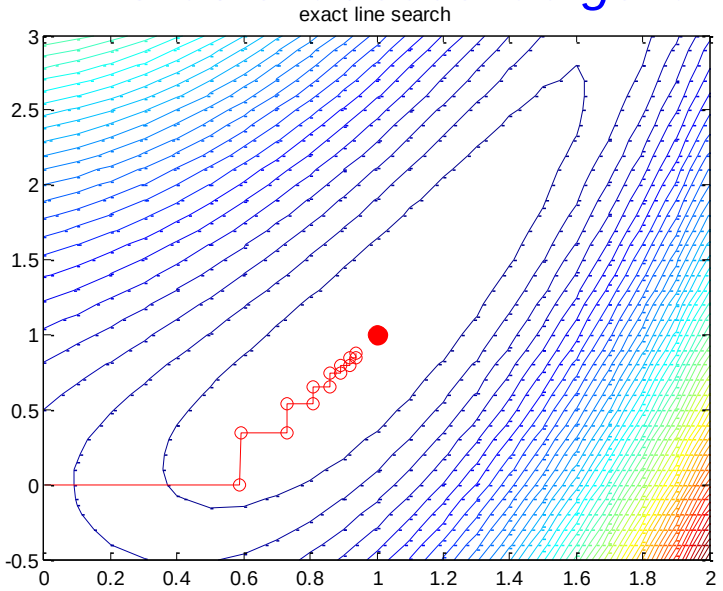
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \boldsymbol{g}_k = \boldsymbol{\theta}_k - \eta_k \nabla f(\boldsymbol{\theta})\big|_k$$

steepestDescentDemo
from Kevin Murphys' PMTK

- Small step size leads to slow convergence and large step size to never converging. In both cases, the step size is kept constant.

Gradient Descent Example

- *Gradient descent algorithm with line search* is shown below.



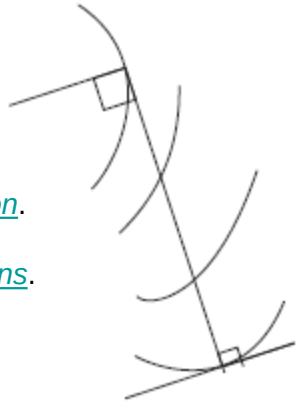
$$f(\boldsymbol{\theta}) = 0.5\ \theta_1^2 - \theta_2\ ^2 + 0.5\ \theta_1 - 1\ ^2$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \boldsymbol{g}_k,\ \boldsymbol{g} = \nabla f$$

$$Pick\ \eta\ to\ minimize: \phi(\eta) = f\ \boldsymbol{\theta}_k + \eta \boldsymbol{d}_k\ \approx f\ \boldsymbol{\theta}_k\ + \eta \nabla f\ \boldsymbol{\theta}_k + \eta \boldsymbol{d}_k\ ^T \boldsymbol{d}_k$$

steepestDescentDemo
from Kevin Murphys' PMTK

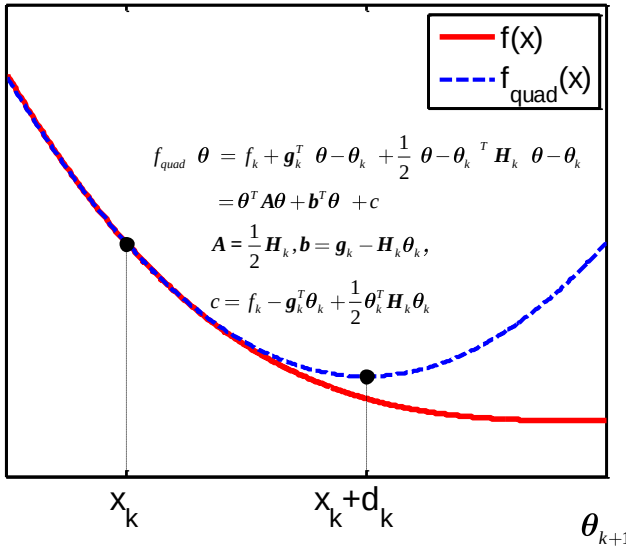
- Bertsekas, D. (1999). *Nonlinear Programming* (Second ed.). Athena Scientific.
- Nocedal, J. and S. Wright (2006). *Numerical Optimization*. Springer.
- Golub, G. and C. F. van Loan (1996). *Matrix computations*. Johns Hopkins University Press.



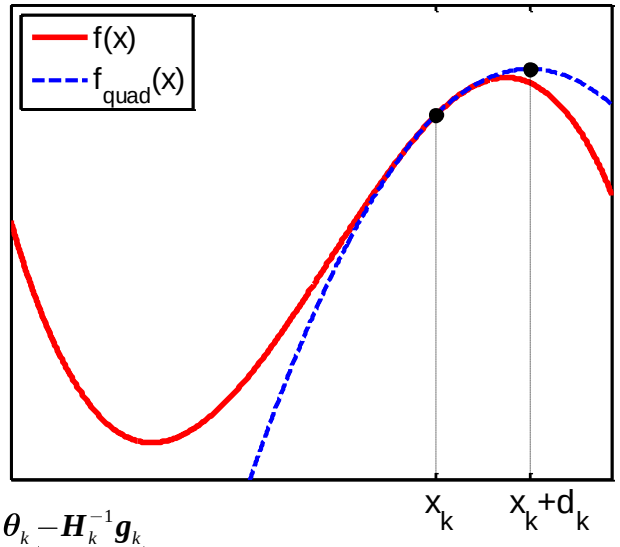
- Consequtive directions are orthogonal: noticing that $\phi'(\eta) = 0$ leads to: $\boldsymbol{d}^T \boldsymbol{g}_{k+1} = 0, \boldsymbol{g}_{k+1} = f'\ \theta + \eta \boldsymbol{d}$. So at the end of each step k, either $\nabla f|_{k+1} = 0$ (stationary point was achieved) or $\boldsymbol{d} \perp \nabla f|_{k+1}$ (i.e. the search stops at a point where the gradient is normal to the search direction).
- To avoid the zig-zag path, modify as: $\theta_{k+1} = \theta_k - \eta_k \boldsymbol{g}_k + \mu_k\ \theta_k - \theta_{k-1}\ , \mu_k \in [0,1]$. This is the so called *heavy-ball method*.
- Alternatively, one can apply the *conjugate gradient method.*

Newton's Method

- Newton's method takes the Hessian into account: $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \boldsymbol{H}_k^{-1} \boldsymbol{g}_k, \boldsymbol{g} = \nabla f$



$$f_{quad}(\theta) = f_k + \boldsymbol{g}_k^T(\theta - \theta_k) + \frac{1}{2}(\theta - \theta_k)^T \boldsymbol{H}_k(\theta - \theta_k)$$
$$= \theta^T A\theta + \boldsymbol{b}^T\theta + c$$
$$A = \frac{1}{2}\boldsymbol{H}_k, \boldsymbol{b} = \boldsymbol{g}_k - \boldsymbol{H}_k\theta_k,$$
$$c = f_k - \boldsymbol{g}_k^T\theta_k + \frac{1}{2}\theta_k^T \boldsymbol{H}_k \theta_k$$



$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k \underbrace{-\boldsymbol{H}_k^{-1}\boldsymbol{g}_k}_{\boldsymbol{d}_k}$$

[newtonsMethodMinQuad](#) and [newtonsMethodNonConvex](#) from [Kevin Murphys' PMTK](#)

- *The step size $d_k = -H_k^{-1} g_k$ is what should be added to θ_k to minimize the 2nd order approximation of f around θ_k.*

$$f'_{quad}(\theta) = 0 \Rightarrow A\theta = -\tfrac{1}{2}\boldsymbol{b} \Rightarrow \boldsymbol{H}_k\theta_{k+1} = -\boldsymbol{g}_k - \boldsymbol{H}_k\theta_k$$
$$\Rightarrow \theta_{k+1} = \theta_k - \boldsymbol{H}_k^{-1}\boldsymbol{g}_k$$

Initialize $\boldsymbol{\theta}_0$;
for $k = 1, 2, \ldots$ *until convergence* **do**
  Evaluate $\mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k)$;
  Evaluate $\mathbf{H}_k = \nabla^2 f(\boldsymbol{\theta}_k)$;
  Solve $\mathbf{H}_k \mathbf{d}_k = -\mathbf{g}_k$ for $\mathbf{d}_k$;
  Use line search to find stepsize η_k along $\mathbf{d}_k$;
  $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta_k \mathbf{d}_k$;

- Vandenberghe, L. (2006). [Applied numerical computing](#): Lecture notes.

- ***H** needs to be pos. definite* (holds for convex functions).
  - If not, $\boldsymbol{d}_k$ may not be a descent direction (Fig. on right).
  - For such cases, use the *Levenberg Marquardt method* (adaptively switch between Newton and Steepest descent updates $\boldsymbol{d}_k = -\boldsymbol{g}_k$)

- Also one can use the *truncated Newton method* (solve $\boldsymbol{H}_k \boldsymbol{d}_k = -\boldsymbol{g}_k$ with CG but truncate the iterations appropriately)

Limited Memory BFGS

- Computing $\boldsymbol{H}$ is expensive. Quasi Newton methods iteratively build approximations using the gradient vector at each step.

- The BFGS method builds an approximation $\boldsymbol{B}$ (rank –two updates) to the Hessian as (we start with $\boldsymbol{B}_0 = \boldsymbol{I}$):

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k} - \frac{\boldsymbol{B}_k \boldsymbol{s}_k \quad \boldsymbol{B}_k \boldsymbol{s}_k}{\boldsymbol{s}_k^T \boldsymbol{B}_k \boldsymbol{s}_k}^T$$

$$\boldsymbol{s}_k = \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$$

$$\boldsymbol{y}_k = \boldsymbol{g}_k - \boldsymbol{g}_{k-1}$$

- Often we provide updates to the inverse of the Hessian:

$$\boldsymbol{C}_{k+1} = \left(\boldsymbol{I} - \frac{\boldsymbol{s}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k}\right) \boldsymbol{C}_k \left(\boldsymbol{I} - \frac{\boldsymbol{s}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k}\right) + \frac{\boldsymbol{s}_k \boldsymbol{s}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k}, \boldsymbol{C}_k \approx \boldsymbol{H}_k^{-1}$$

- To avoid storing the Hessian or its inverse, *limited memory BFGS (L-BFGS)* is used where $\boldsymbol{H}_k^{-1} \boldsymbol{g}_k$ is performed by a sequence of inner products with s_k and $\boldsymbol{y}_k$ using only the m most recent (~ 20) pairs of (s_k, y_k). This requires only $\mathcal{O}(mD)$ storage vs $\mathcal{O}(D^2)$ for the full Hessian.

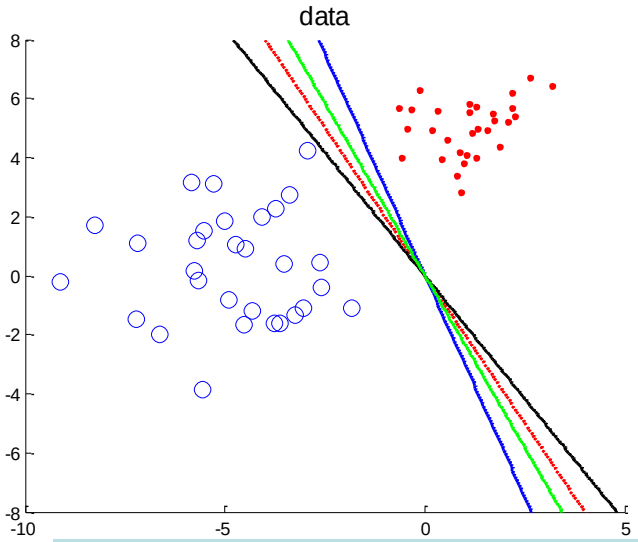
  - Nocedal, J. and S. Wright (2006). *Numerical Optimization*. Springer.

L_2 Regularization

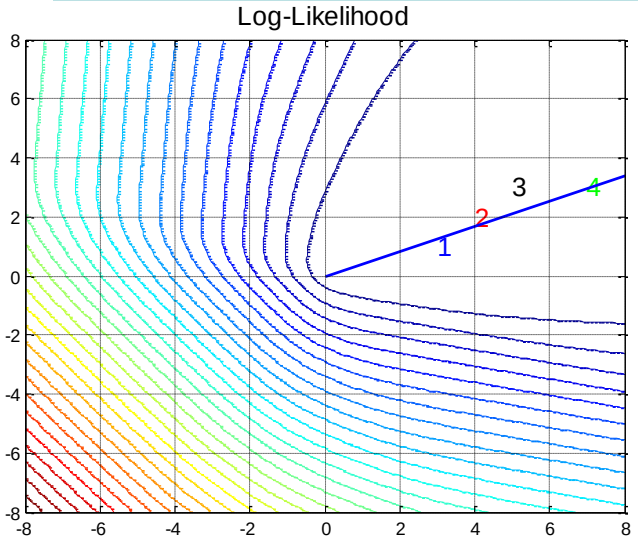
- Regularization is important for classification even with lots of data.
- *For linearly separable data, the MLE is obtained by* $||\mathbf{w}|| \rightarrow \infty$*, corresponding to an infinitely steep sigmoid.* In the two class linearly separable data, we show the Log-likelihood.
- The line is drawn from the origin in the direction of the MLE (which is at ∞). The numbers correspond to the lines on the top Fig.
- This solution is very brittle and does not generalize well.
- To prevent this, we use L_2 regularization.

$$f(\mathbf{w}) = NLL(\mathbf{w}) + \lambda\ \mathbf{w}^T\mathbf{w}\ \textit{(objective function)}$$
$$\mathbf{g}(\mathbf{w}) = \mathbf{g}(\mathbf{w}) + \lambda\,\mathbf{w}\ \textit{(gradient)}$$
$$\mathbf{H}(\mathbf{w}) = \mathbf{H}(\mathbf{w}) + \lambda\,\mathbf{I}\ \textit{(Hessian)}$$

- These modifications provide no difficulty with any gradient-based optimization toolbox.



logregLaplaceGirolamiDemo from Kevin Murphys' PMTK



Iterative Reweighted Least Squares

- In linear regression models, the MLE for a Gaussian noise model had a closed-form solution. This was a consequence of the quadratic dependence of the log likelihood function on $\boldsymbol{w}$.
- For logistic regression, there is no closed-form solution due to the nonlinearity of the logistic sigmoid function.
- However, the error function is convex and hence has a unique minimum.
- In addition, it can be minimized by an iterative technique based on *Newton-Raphson, using a quadratic approximation to the log likelihood function.*
- The *Newton-Raphson update*, for minimizing a function $E(\boldsymbol{w})$, takes the form

$$\boldsymbol{w}^{(new)} = \boldsymbol{w}^{(old)} - \boldsymbol{H}^{-1}\nabla E(\boldsymbol{w}), \;\; \boldsymbol{H}_{ij} = \partial^2 E(\boldsymbol{w}) \big/ \partial w_i w_j \;\; (Hessian)$$

Iterative Reweighted Least Squares

- Let us first apply the Newton-Raphson method to the linear regression model with the sum-of-squares error function. The gradient and Hessian of this error function are given by

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right)^2 \Rightarrow \nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n) - t_n\right)\boldsymbol{\phi}_n = \boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{\Phi}^T\boldsymbol{t}$$

  where $\boldsymbol{\Phi}$ is the $N \times M$ matrix whose n^{th} row is given by $\boldsymbol{\phi}_n^T$.

- The Newton-Raphson update takes the form:

$$\boldsymbol{w}^{(new)} = \boldsymbol{w}^{(old)} - \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w}^{(old)} - \boldsymbol{\Phi}^T\boldsymbol{t}\right) = \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{t}$$

$$where\ \boldsymbol{H} = \nabla\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T = \boldsymbol{\Phi}^T\boldsymbol{\Phi}$$

- This is the standard least-squares solution to *linear regression. The error function is quadratic and hence the Newton-Raphson gives the exact solution in one step!*

Iterative Reweighted Least Squares

- We similarly apply the Newton-Raphson to the cross-entropy error function for the logistic regression model:

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{t} \mid \boldsymbol{w}) = -\sum_{n=1}^{N} \left\{ t_n \ln y_n + (1-t_n) \ln \left(1-y_n\right) \right\}, \; y_n = \sigma(a_n), \; a_n = \boldsymbol{w}^T \boldsymbol{\phi}_n \Rightarrow$$

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} \left(y_n - t_n\right) \boldsymbol{\phi}_n = \boldsymbol{\Phi}^T \left(\boldsymbol{y} - \boldsymbol{t}\right)$$

$$\boldsymbol{H} = \nabla\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} y_n \left(1-y_n\right) \boldsymbol{\phi}_n \boldsymbol{\phi}_n^T = \boldsymbol{\Phi}^T \boldsymbol{R} \boldsymbol{\Phi}, \; \boldsymbol{R} = diag\left(y_1\left(1-y_1\right), ..., y_n\left(1-y_n\right)\right)$$

  where in the last equation we used $d\sigma / da = \sigma(1-\sigma)$

- The Hessian now depends on $\boldsymbol{w}$ through *the weighting matrix* $\boldsymbol{R}$, since the error function is no longer quadratic.
- Using $0 < y_n < 1$, which follows from the form of the logistic sigmoid function, we see that $\boldsymbol{u}^T \boldsymbol{H} \boldsymbol{u} > 0$ for any $\boldsymbol{u}$, and so $\boldsymbol{H}$ is positive definite.
- It follows that the error function is a convex function of $\boldsymbol{w}$ and hence has a unique minimum.

Iterative Reweighted Least Squares

- The Newton-Raphson update then becomes:

$$\boldsymbol{w}^{(new)} = \boldsymbol{w}^{(old)} - \left(\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T (\boldsymbol{y} - \boldsymbol{t}) = \left(\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1} \left\{\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{w}^{(old)} - \boldsymbol{\Phi}^T (\boldsymbol{y} - \boldsymbol{t})\right\}$$

$$= \left(\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{z}, \text{ where } \boldsymbol{z} = \boldsymbol{\Phi}\boldsymbol{w}^{(old)} - \boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{t}\right)$$

- The update formula takes the form of a set of normal equations for a weighted least-squares problem.
- *Because $\boldsymbol{R}$ depends on $\boldsymbol{w}$, we apply the normal equations iteratively.*
- For this reason, the algorithm is known as iterative reweighted least squares - IRLS.

$\boldsymbol{w} = \mathbf{0}_D;$

$w_0 = \log\ \bar{t} / (1 - \bar{t})\ ;$

repeat

$\eta_i = w_0 + \boldsymbol{w}^T \boldsymbol{x}_i\ ;$

$y_i = sigm(\eta_i);$

$r_i = y_i(1 - y_i);$

$z_i = h_i + (t_i - y_i) / r_i;$

$\boldsymbol{R} = diag(r_{1:N});$

$\boldsymbol{w} = \boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}^{-1}\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{z};$

until *converged;*

- Rubin, D. B. (1983). Iteratively reweighted least squares. In *Encyclopedia of Statistical Sciences*, Volume 4, pp. 272–275. Wiley.

Iterative Reweighted Least Squares

$$\boldsymbol{w}^{(new)} = \left(\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{z}, \text{where } \boldsymbol{z} = \boldsymbol{\Phi}\boldsymbol{w}^{(old)} - \boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{t}\right),$$

$$\boldsymbol{R} = diag\left(y_1\left(1-y_1\right), ..., y_n\left(1-y_n\right)\right)$$

- The elements of the diagonal weighting matrix $\boldsymbol{R}$ can be interpreted as variances.

- Indeed note that the mean and variance of t in the logistic regression model are given as:

$$\mathbb{E}\left[t\right] = 1 \times \sigma(\boldsymbol{x}) + 0 \times \left(1 - \sigma(\boldsymbol{x})\right) = y$$

$$\text{var}\left[t\right] = \mathbb{E}\left[t^2\right] - \mathbb{E}\left[t\right]^2 = \sigma(\boldsymbol{x}) - \sigma(\boldsymbol{x})^2 = y(1-y)$$

where we used $t^2 = t$ for $t \in \{0, 1\}$.

Iterative Reweighted Least Squares

$$\boldsymbol{w}^{(new)} = \left(\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{z}, \text{where } \boldsymbol{z} = \boldsymbol{\Phi}\boldsymbol{w}^{(old)} - \boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{t}\right),$$

$$\boldsymbol{R} = diag\left(y_1\left(1-y_1\right), \ldots, y_n\left(1-y_n\right)\right)$$

- We can also interpret *IRLS* as the solution to a linearized problem in the space of the variable $a = \boldsymbol{w}^T\boldsymbol{\phi}$.

- The quantity z_n, which corresponds to the n^{th} element of $\boldsymbol{z}$, can be seen as ***an effective target value*** in this space obtained by making a local linear approximation to the logistic sigmoid function around $\boldsymbol{w}^{(\boldsymbol{old})}$.

$$y_n = \sigma(a_n) = \frac{1}{1+e^{-a_n}}, a_n = \boldsymbol{w}^T\boldsymbol{\phi}_n, \frac{dy_n}{da_n} = y_n\left(1-y_n\right) \Rightarrow$$

$$t_n \cong y_n + \frac{dy_n}{da_n}\Big|_{w^{(old)}}\left(a_n(w) - a_n(w^{(old)})\right)$$

$$a_n(w) \simeq a_n(w^{(old)}) + \frac{da_n}{dy_n}\Big|_{w^{(old)}}(t_n - y_n) = \phi_n^T w^{(old)} - \frac{(y_n - t_n)}{y_n(1-y_n)} = z_n$$

Multiclass Logistic Regression

- For $K > 2$ classes, the posterior probabilities are given by a *softmax transformation of linear functions of the feature variables* as follows:

$$p(C_k \mid \boldsymbol{\phi}) = y_k(\boldsymbol{\phi}) = \frac{e^{a_k}}{\sum_j e^{a_j}}, \quad a_k = \ln\left(p(\boldsymbol{\phi} \mid C_k)\, p(C_k) \right) = \boldsymbol{w}_k^T \boldsymbol{\phi}$$

- Use maximum likelihood to determine $\{\boldsymbol{w}_k\}$ for this model.

- We need to compute the derivatives of y_k with respect to all of the activations a_j. They are given as:

$$y_k(\boldsymbol{\phi}) = \frac{e^{a_k}}{\sum_j e^{a_j}} \Rightarrow \frac{\partial y_k}{\partial a_j} = y_k(\delta_{kj} - y_j)$$

- Krishnapuram, B., L. Carin, M. Figueiredo, and A. Hartemink (2005). Learning sparse bayesian classifiers: multi-class formulation, fast algorithms, and generalization bounds. *IEEE Transaction on Pattern Analysis and Machine Intelligence*.

Multiclass Logistic Regression

- Next we write down the likelihood function.
- We use the 1-of-K coding scheme. The target $\boldsymbol{t}_n$ for a feature $\boldsymbol{\phi}_n$ belonging to class C_k is a binary vector with all elements zero except for element k, which equals one.
- The likelihood function is then given by

$$p(\boldsymbol{T} \mid \boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k \mid \boldsymbol{\phi}_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}},$$

$$y_{nk} = y_k(\boldsymbol{\phi}_n), \boldsymbol{T} \; N \times K \text{ matrix}, T_{nk} = t_{nk}$$

- The negative log likelihood is then

$$E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = -\ln p(\boldsymbol{T} \mid \boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

- This is the cross-entropy error function for the multiclass classification.

Multiclass Logistic Regression

$$E(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_K) = -\ln p(\boldsymbol{T} \mid \boldsymbol{w}_1,\ldots,\boldsymbol{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

- We now take the gradient of the error function with respect to one of the $\boldsymbol{w}_j$.
- Using $y_{nk} = \dfrac{e^{a_{kn}}}{\sum_j e^{a_{jn}}}$, $\log y_{nk} = a_{kn} - \log \sum_j e^{a_{jn}}$, $a_{kn} = \boldsymbol{w}_k^T \boldsymbol{\phi}_n$ and $\sum_k t_{nk} = 1$, we obtain

$$\nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_K) = \sum_{n=1}^{N} \left(y_{nj} - t_{nj} \right) \boldsymbol{\phi}_n$$

- This is again the form for the gradient derived earlier for the sum-of-squares error function with the linear regression model and the cross-entropy error for logistic regression.
- We *use a sequential algorithm*

$$\boldsymbol{w}_j^{(\tau+1)} = \boldsymbol{w}_j^{(\tau)} - \eta \nabla E_{nj} = \boldsymbol{w}^{(\tau)} - \eta (y_{nj} - t_{nj}) \boldsymbol{\phi}_n$$

General Form: Derivative of log Likelihood

- We have seen that the derivative of the log likelihood function for a linear regression model with respect to the parameter vector $\boldsymbol{w}$ for a data point n took the form of the 'error' $y_n - t_n$ times the feature vector $\boldsymbol{\phi}_n$.

- Similarly, for the combination of logistic sigmoid activation function and cross-entropy error function

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{t} \mid \boldsymbol{w}) = -\sum_{n=1}^{N} \left\{ t_n \ln y_n + (1 - t_n) \ln \left(1 - y_n\right) \right\},$$

  and for the softmax activation function with the multiclass cross-entropy error function

$$E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = -\ln p(\boldsymbol{T} \mid \boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

- All these are examples of a more general result.

IRLS Algorithm for the Multiclass Problem

- To find a batch algorithm, we again appeal to the Newton-Raphson update to obtain the corresponding IRLS algorithm for the multiclass problem.

- This requires evaluation of the Hessian matrix that comprises blocks of size $M \times M$ in which block j, k is given by

$$\nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = \sum_{n=1}^{N} \left(y_{nj} - t_{nj} \right) \boldsymbol{\phi}_n \Rightarrow \nabla_{\boldsymbol{w}_k} \nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = \sum_{n=1}^{N} y_{nk} \left(\delta_{kj} - y_{nj} \right) \boldsymbol{\phi}_n \boldsymbol{\phi}_n^T$$

where we used:

$$\nabla_{\boldsymbol{w}_k} y_{nj} = \frac{dy_{nj}}{da_{nk}} \nabla_{\boldsymbol{w}_k} a_{nk} = y_{nk} \left(\delta_{kj} - y_{nj} \right) \boldsymbol{\phi}_n$$

- As with the two-class problem, the Hessian matrix for the multiclass logistic regression model is positive definite and so the error function again has a unique minimum.

- Bishop, C. M. and I. T. Nabney (2008). *Pattern Recognition and Machine Learning: A Matlab Companion*. Springer. In preparation

MAP Estimate

- One can define an appropriate prior

$$p(\boldsymbol{W}) = \prod_c \mathcal{N}(\boldsymbol{w}_c \mid \boldsymbol{0}, \boldsymbol{V}_0)$$

- Using this, the modified negative log-likelihood and its gradient and Hessian are:

$$f'(\boldsymbol{W}) = -\log p(\mathcal{D} \mid \boldsymbol{W}) - \log p(\boldsymbol{W}) = f(\boldsymbol{W}) + \frac{1}{2}\sum_c \boldsymbol{w}_c \boldsymbol{V}_0^{-1} \boldsymbol{w}_c$$

$$\boldsymbol{g}'(\boldsymbol{W}) = \boldsymbol{g}(\boldsymbol{W}) + \boldsymbol{V}_0^{-1}\sum_c \boldsymbol{w}_c$$

$$\boldsymbol{H}'(\boldsymbol{W}) = \boldsymbol{H}(\boldsymbol{W}) + \boldsymbol{I}_C \otimes \boldsymbol{V}_0^{-1} \left((CD)\times(CD)\right)$$

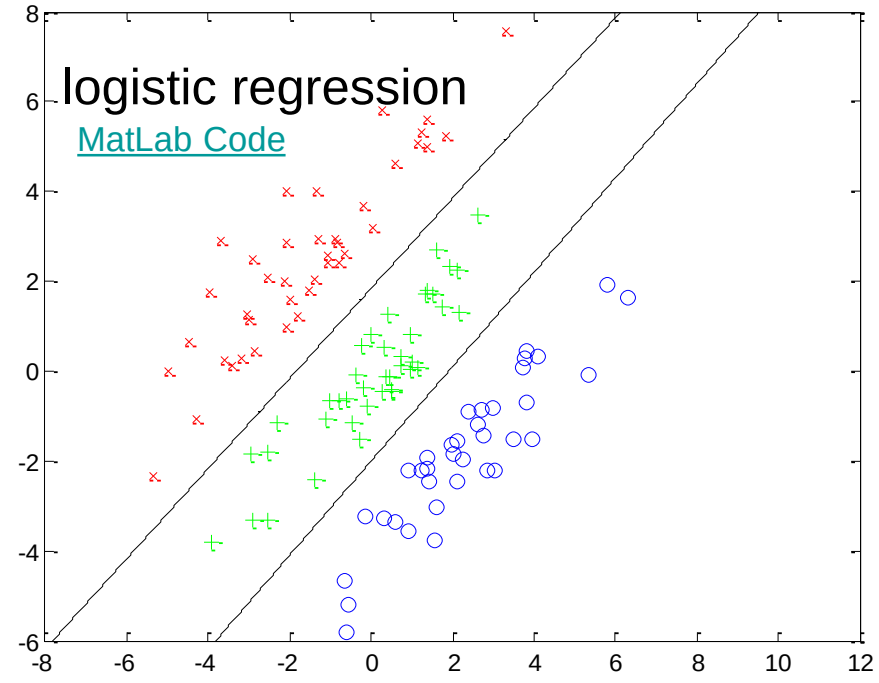
logregFit
from Kevin Murphys' PMTK

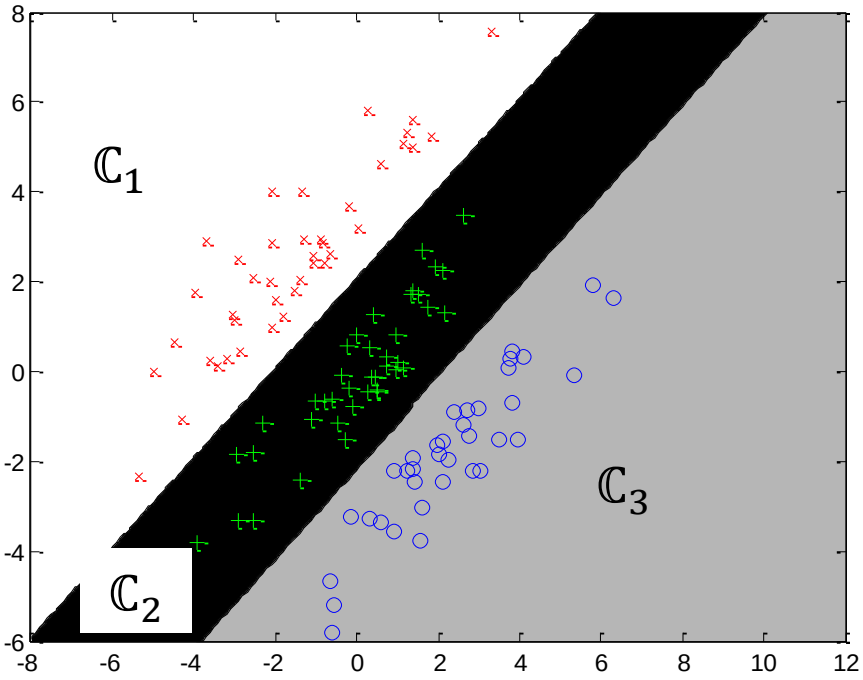
- The symbol $\boldsymbol{I}_C \otimes \boldsymbol{V}_0^{-1}$ is used here to denote Kronecker matrix product.

- One can apply *limited memory BFGS* to find the MAP estimate.

Multiclass Logistic Regression Example

This is the same example of 3 class classification discussed earlier.



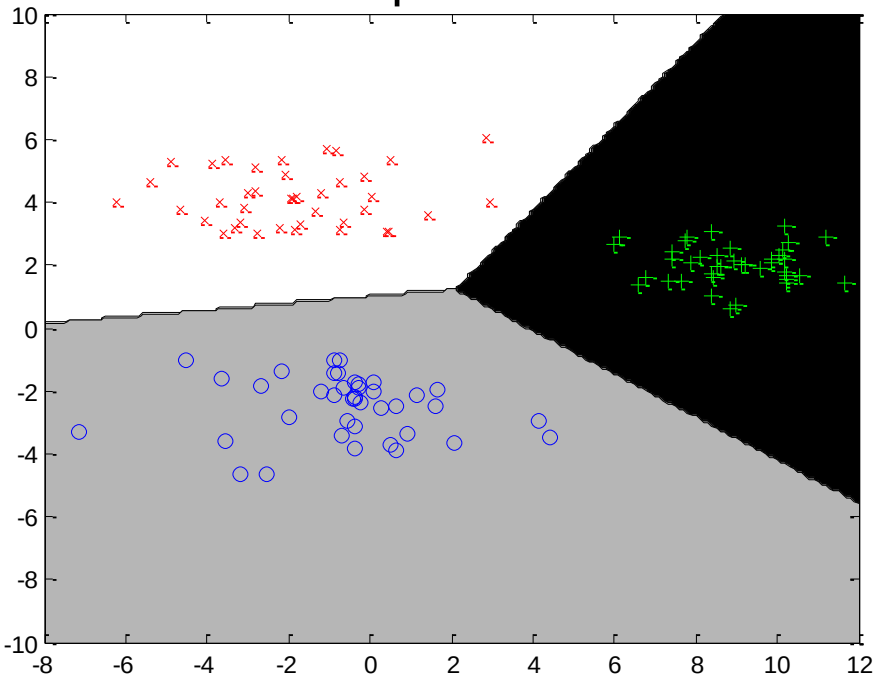


Here, a_1, a_2, a_3 are the feature functions computed by following the logistic regression algorithm. Region C_1, C_2, C_3 in the right plot can be interpreted as:

$\mathbb{C}_m: p(C_m|\phi) = \max\left(p(C_1|\phi), p(C_2|\phi), p(C_3|\phi)\right)$ where $p\left(C_k|\phi\right) = \dfrac{\exp\left(a_k\right)}{\sum_j \exp\left(a_j\right)}$

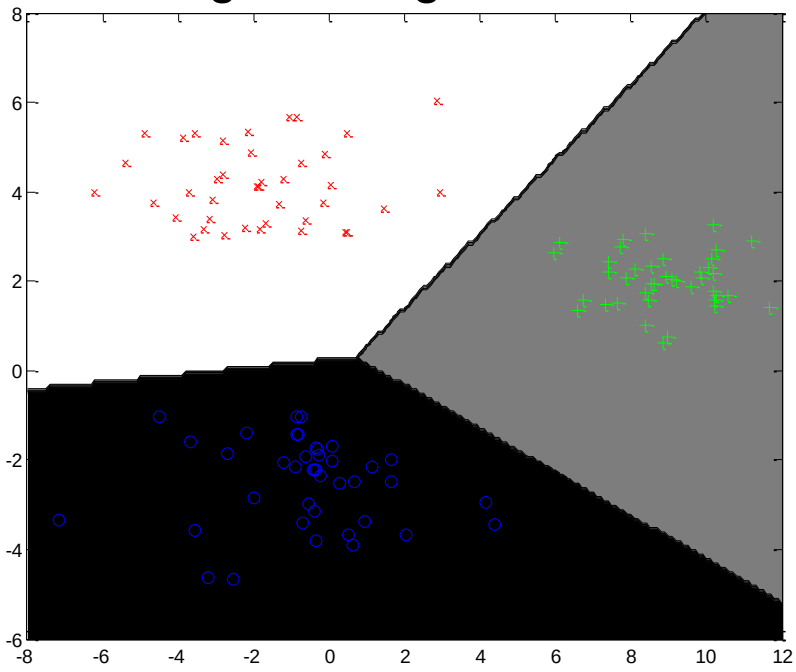
Multiclass Logistic Regression Example

Least-squares



MatLab Code:
Use 'data_3c_2'

Logistic Regression



MatLab Code:
Use 'data_3c_2'